Relenv

Daniel A. Wozniak

Sep 09, 2023

CONTENTS

Relenv creates re-producable and re-locatable python builds. The builds created with Relenv are re-producable in the sense that all binaries for the builds are built from source. These builds are re-locatable meaning you can move the root directory around on the filesystem; and even different machines of same architecture.

Relenv can be thought of in a similar way to projects like virtualenv and pyenv. The difference between Relenv and these projects is that each relenv contains the entire python interpereter. Relenv environments can be moved around and shared with others.

CHAPTER

ONE

USEFUL LINKS

- Documentation
- Issue Tracker
- Source Code
- Python Packages

CHAPTER

TWO

TOPICS

2.1 Installation

You can install relenv like any other python package using pip. Using a virtual environment is recommended.

```
pip install relenv
```

2.2 Usage Overview

After installing relenv, you will have access to its CLI. You can see all supported commands by running the following...

```
relenv --help
```

The most common sub command of Relenv is the *Create* command. Create is used to create new relenv environments. When using relenv for the first time you will need to *Build* or *Fetch* pre-built Python build.

2.2.1 Fetch

The fetch command can be used to download a pre-built Relenv Python.

```
relenv fetch
```

Now that you have a base Relenv Python build, you can use the *Create* command to make new Relenv environments. See the full *fetch* documentation for more details.

2.2.2 Build

The build command is what relenv uses to build a Relenv Python environment.

```
relenv build
```

See the full *build* documentation for more details.

2.2.3 Create

Use create to make a new relenv environment.

```
relenv create myenv
```

The new 'myenv' environment is fully self contained. You can pip install any python packages you would like into the new 'myenv' environment. Everything will be installed into myenv's site-packages directory. Any scripts created by pip will use myenv's Python interpreter. See the full *create* documentation for more details.

```
myenv/bin/pip3 install mycoolpackage
```

2.3 Additional Dependencies (Linux)

Some python libraries do not provide wheels and require additional libraries to install properly. You can handle installing these python packages in two ways. You build them using system dependencies or you can install the needed depenency libraries into a relenv environment.

The general procedure for installing python modules to use your system's libraries is to install the required sytem packages which contain the header files needed for the package. Then using your system's compiler configured with the system include path and system librariy directory. This is the default behavior when using pip to install to relenv.

To install additional libraries into the relenv environment you will compile the library from source using the relenv toolchain compiler. Relenv provides the `relenv buildenv` to help simplify setting up your environment to use the relenv toolchain.Link the library against relenv's library directory and setting the rpath to relenvs' library directory. Then run `relenv check` to check and potentially fix the binary's rpath, making it relative. Finally using pip to install the intended python library.

2.3.1 Installing pycurl Using System Libraries

This is an example of installing pycurl using the system's libcurl on Debian Linux.

```
relenv create myenv
sudo apt-get install libcurl4-openssl-dev
myenv/bin/pip3 install pycurl --no-cache
```

2.3.2 Installing pygit2 Using System Libraries

This is an example of installing pygit2 using the system's libgit2 on Debian Linux.

```
relenv create myenv
sudo apt-get install libgit2-dev libssh2-1-dev
myenv/bin/pip3 install libgit2 --no-binary=":all:"
```

2.3.3 Installing python-ldap Using System Libraries

This is an example of installing python-ldap using the system's open-ldap on Debian Linux.

```
relenv create myenv
sudo apt-get install openldap-dev libsasl2-dev
myenv/bin/pip3 install python-ldap
```

2.3.4 Installing M2Crypto Using System Libraries

This is an example of installing M2Crytpo using the system's openssl on Debian Linux.

```
relenv create myenv
sudo apt-get install libssl-dev
env LDFLAGS="-L/usr/lib" \
   CFLAGS="-I/usr/include" \
   SWIG_FEATURES="-I/usr/include" \
   myenv/bin/pip3 install m2crypto
```

2.3.5 Building and Installing curl For pycurl

In this example, we use `relenv buildenv` to setup our environment. Install curl after building it from source. Run `relenv check` to fix the rpaths, making them relative. Then installing pycurl using the relenv's pip.

```
relenv create myenv
# C extensions require a toolchain on linux
relenv toolchain fetch
# Load some useful build variables into the environment
source <(myenv/bin/relenv buildenv)
wget https://curl.se/download/curl-8.0.1.tar.gz
tar xgf curl-8.0.1.tar.gz
cd curl-8.0.1
# Configure curl using the build environment.
./configure --prefix=$RELENV_PATH --with-openssl=$RELENV_PATH
make
make install
cd ..
# Install pycurl, adjust the path so pycurl can find the curl-config executable
PATH="${RELENV_PATH}/bin:${PATH}" myenv/bin/pip3 install pycurl
```

2.3.6 Building and Installing libgit2 for pygit2

In this example we use Cmake to build and install libssh2 and libgit2, pre-requsits for pygit2.

```
relenv create myenv
# C extensions require a toolchain on linux
relenv toolchain fetch
# Load some useful build variables into the environment
source <(myenv/bin/relenv buildenv)

```

(continues on next page)

(continued from previous page)

```
# Build and install libssh2
wget https://www.libssh2.org/download/libssh2-1.10.0.tar.gz
tar xvf libssh2-1.10.0.tar.gz
cd libssh2-1.10.0
mkdir bin
cd bin
cmake .. \
  -DENABLE_ZLIB_COMPRESSION=ON \
  -DOPENSSL_ROOT_DIR="$RELENV_PATH" \
  -DBUILD_SHARED_LIBS=ON \
  -DBUILD_EXAMPLES=OFF \
  -DBUILD_TESTING=OFF \
  -DCMAKE_INSTALL_PREFIX="$RELENV_PATH"
cmake --build .
cmake --build . --target install

cd ../..

# Build and install libssh2 (version 0.5.x for pygit2)
wget https://github.com/libgit2/libgit2/archive/refs/tags/v0.5.2.tar.gz
tar xvf v0.5.2.tar.gz
cd libgit2-0.5.2
mkdir build
cd build
cmake ..  \
  -DOPENSSL_ROOT_DIR="$RELENV_PATH" \
  -DBUILD_CLI=OFF \
  -DBUILD_TESTS=OFF \
  -DUSE_SSH=ON \
  -DCMAKE_INSTALL_PREFIX="$RELENV_PATH"
cmake --build .
cmake --build . --target install
cd ../..

# Run relenv check
myenv/bin/relenv check

myenv/bin/pip3 install pygit2 --no-binary=":all:"
```

2.3.7 Building and Installing open-ldap For python-ldap

In this example, we use `relenv buildenv` to setup our environment. Build and install sasl and open-ldap. Run `relenv check` to fix the rpaths, making them relative. Then install python-ldap using the relenv's pip.

```
relenv create myenv
# C extensions require a toolchain on linux
relenv toolchain fetch
# Load some useful build variables into the environment
source <(myenv/bin/relenv buildenv)

# Build and Install sasl
```

(continues on next page)

(continued from previous page)

```
wget https://github.com/cyrusimap/cyrus-sasl/releases/download/cyrus-sasl-2.1.28/cyrus-
↪sasl-2.1.28.tar.gz
tar xvf cyrus-sasl-2.1.28.tar.gz
cd cyrus-sasl-2.1.28
./configure --prefix=$RELENV_PATH
make
make install
cd ..

# Build and Install Open LDAP
wget https://www.openldap.org/software/download/OpenLDAP/openldap-release/openldap-2.5.
↪14.tgz
tar xvf openldap-2.5.14.tgz
cd openldap-2.5.14
./configure --prefix=$RELENV_PATH
make
make install
cd ..

# Fix any non-relative rpaths
myenv/bin/relenv check

myenv/bin/pip3 install python-ldap
```

2.3.8 Installing M2Crypto Using Relenv's Libraries

This is an example of installing M2Crytpo using the relenvs's openssl on Debian Linux.

```
relenv create myenv
source <(myenv/bin/relenv buildenv)
SWIG_FEATURES="-I${RELENV_ROOT}/include" myenv/bin/pip3 install m2crypto
```

2.4 Relenv's CLI

Relenv exposes the following subcommands.

The recommended way to run relenv commands is to use the exposed commands.

```
relenv <subcommand> <args>
```

Note: You can suppress CLI output by setting `CI` to any value in your environment.

2.4.1 relenv

```
relenv
```

Options

Relenv

```
usage: relenv [-h] [--version]
              {build,toolchain,create,fetch,check,buildenv} ...
```

Named Arguments

--version show program's version number and exit

Sub-commands

build

Build Relenv Python Environments from source

```
relenv build [-h] [--arch {x86_64,aarch64}] [--clean]
             [--python {3.10.13,3.11.5}] [--no-cleanup] [--force-download]
             [--step STEP] [--check-versions]
```

Named Arguments

--arch Possible choices: x86_64, aarch64

The host architecture [default: "x86_64"]

Default: "x86_64"

--clean Clean up before running the build. This option will remove the logs, src, build, and previous tarball.

Default: False

--python Possible choices: 3.10.13, 3.11.5

The python version [default: "3.10.13"]

Default: "3.10.13"

--no-cleanup By default the build directory is removed after the build tarball is created. Setting this option will leave the build directory in place.

Default: False

--force-download Force downloading source tarballs even if they exist

Default: False

--step A step to run alone, can use multiple of this argument. When this option is used to invoke builds, depenencies of the steps are ignored. This option should be used with care, as it's easy to request a situation that has no chance of being succesful.

Default: []

--check-versions Check for new version of python and it's depenencies, then exit.

Default: False

toolchain

Build Linux Toolchains

```
relenv toolchain [-h] [--arch {x86_64,aarch64}] [--clean] [--crosstool-only]
                 {build,fetch}
```

Positional Arguments

command Possible choices: build, fetch

What type of toolchain operation to perform: build or fetch

Default: "fetch"

Named Arguments

--arch Possible choices: x86_64, aarch64

Architecture to build or fetch

Default: "x86_64"

--clean Whether or not to clean the toolchain directories

Default: False

--crosstool-only When building only build Crosstool NG. Do not build toolchains

Default: False

create

Create a Relenv environment. This will create a directory of the given name with newly created Relenv environment.

```
relenv create [-h] [--arch {x86_64,aarch64}] [--python {3.10.13,3.11.5}] name
```

Positional Arguments

name The name of the directory to create

Named Arguments

--arch Possible choices: x86_64, aarch64

The host architecture [default: “x86_64”]

Default: “x86_64”

--python Possible choices: 3.10.13, 3.11.5

The python version [default: “3.10.13”]

Default: “3.10.13”

fetch

Fetch relenv builds

```
relenv fetch [-h] [--arch {x86_64,aarch64}] [--python PYTHON]
```

Named Arguments

--arch Possible choices: x86_64, aarch64

Architecture to download. [default: “x86_64”]

Default: “x86_64”

--python The python version [default: “3.10.13”]

Default: “3.10.13”

check

Check relenv integrity

```
relenv check [-h]
```

buildenv

Relenv build environment

```
relenv buildenv [-h]
```

2.4.2 build

Relenv build is resposible for building Python and it's dependencies from source. The build process also ensures Python is re-locatable. The directory containg the python build can be moved around on the filesystem or to another host machine of the same architecture.

```
relenv build
```

Options

Build Relenv Python Environments from source

```
usage: relenv build [-h] [--arch {x86_64,aarch64}] [--clean]
                    [--python {3.10.13,3.11.5}] [--no-cleanup]
                    [--force-download] [--step STEP] [--check-versions]
```

Named Arguments

--arch Possible choices: x86_64, aarch64

The host architecture [default: "x86_64"]

Default: "x86_64"

--clean Clean up before running the build. This option will remove the logs, src, build, and previous tarball.

Default: False

--python Possible choices: 3.10.13, 3.11.5

The python version [default: "3.10.13"]

Default: "3.10.13"

--no-cleanup By default the build directory is removed after the build tarball is created. Setting this option will leave the build directory in place.

Default: False

--force-download Force downloading source tarballs even if they exist

Default: False

--step A step to run alone, can use multiple of this argument. When this option is used to invoke builds, depenencies of the steps are ignored. This option should be used with care, as it's easy to request a situation that has no chance of being succesful.

Default: []

--check-versions Check for new version of python and it's depenencies, then exit.

Default: False

2.4.3 fetch

```
relenv fetch
```

Options

Fetch relenv builds

```
usage: relenv fetch [-h] [--arch {x86_64,aarch64}] [--python PYTHON]
```

Named Arguments

--arch Possible choices: x86_64, aarch64

Architecture to download. [default: “x86_64”]

Default: “x86_64”

--python The python version [default: “3.10.13”]

Default: “3.10.13”

2.4.4 create

```
relenv create
```

Options

Create a Relenv environment. This will create a directory of the given name with newly created Relenv environment.

```
usage: relenv create [-h] [--arch {x86_64,aarch64}]
                     [--python {3.10.13,3.11.5}]
                     name
```

Positional Arguments

name The name of the directory to create

Named Arguments

--arch Possible choices: x86_64, aarch64

The host architecture [default: "x86_64"]

Default: "x86_64"

--python Possible choices: 3.10.13, 3.11.5

The python version [default: "3.10.13"]

Default: "3.10.13"

2.4.5 `toolchain`

```
relenv toolchain
```

Options

Build Linux Toolchains

```
usage: relenv toolchain [-h] [--arch {x86_64,aarch64}] [--clean]
                        [--crosstool-only]
                        {build,fetch}
```

Positional Arguments

command Possible choices: build, fetch

What type of toolchain operation to perform: build or fetch

Default: "fetch"

Named Arguments

--arch Possible choices: x86_64, aarch64

Architecture to build or fetch

Default: "x86_64"

--clean Whether or not to clean the toolchain directories

Default: False

--crosstool-only When building only build Crosstool NG. Do not build toolchains

Default: False

Comprehensive CLI documentation can be found *here*.

2.5 Toolchains

Relenv uses toolchains to compile Python (and it's dependencies) on Linux platforms. These toolchains consist of GCC, Binutils, and GLibc and are built using crosstool-ng. Relenv's toolchains are pre-built. Users of Relenv will only need a toolchain when installing C extensions which need to re-main portable accross multiple Linux OSes. When working with pure python applications users should not need to concern themselves with toolchains.

2.5.1 Building Toolchains

Building toolchains is a farily expensive and lengthy process. It's recommended that you have 16GB of RAM and 40GB of free disk space. The example below is using Centos Stream 8.

```
sudo yum -y groupinstall "Development Tools"
sudo yum -y --enablerepo=powertools install vim python3 texinfo help2man ncurses-devel
```

Running Relenv's toolchain build command will do the following

- Download crosstool-ng
- Configure and compile crosstool-ng
- Use Relenv's crosstool config files to compile the requests architectures

```
git clone git@github.com:saltstack/relenv.git
cd relenv
python3 -m relenv toolchain build --arch=x86_64 --arch=aarch64
```

2.5.2 Pre-Built Toolchains

Under most circumstances using a pre-built toolchain is preffered over building the toolchain yourself.

```
python3 -m relenv toolchain fetch --arch=x86_64
```

2.6 Contributing

Relenv is open source, and welcomes contributions from the community.

2.6.1 Getting the Code

The code is available under the saltstack organization on github. We use the fork and branch workflow, so once you've created your own fork, go ahead and clone it to start hacking!

```
git clone git@github.com:<username>/relenv.git
```

2.7 Relenv's Developer API

Relenv has no stable programable API and is only intended to be used from the command line. This is documentation on the internals of relenv.

Note: There is no public API for relenv. This documentation is intended for those developing on, and contributing to, relenv.

2.7.1 Entrypoint

The entrypoint into relenv.

`relenv.__main__.main()`

Run the relenv cli and disbatch to subcommands.

`relenv.__main__.setup_cli()`

Build the argparser with its subparsers.

The modules with commands to add must specify a setup_parser function that takes in the subparsers object from *argparse.add_subparsers()*

Returns
The fully setup argument parser

Return type
`argparse.ArgumentParser`

2.7.2 Common Code

Common classes and values used around relenv.

exception `relenv.common.RelenvException`

Base class for exeptions generated from relenv.

class `relenv.common.WorkDirs(`*root*`)`

Simple class used to hold references to working directories relenv uses relative to a given root.

Parameters
root (*str*) – The root of the working directories tree

`relenv.common.archived_build(`*triplet=None*`)`

Finds a the location of an archived build.

Parameters
triplet (*str*) – The build triplet to find

Returns
The location of the archived build

Return type
`pathlib.Path`

`relenv.common.build_arch()`

Return the current machine.

relenv.common.**check_url**(*url*, *timeout=30*)

Check that the url returns a 200.

relenv.common.**download_url**(*url*, *dest*, *verbose=True*, *backoff=3*, *timeout=60*)

Download the url to the provided destination.

This method assumes the last part of the url is a filename. (https://foo.com/bar/myfile.tar.xz)

Parameters

- **url** (*str*) – The url to download
- **dest** (*str*) – Where to download the url to
- **verbose** (*bool*) – Print download url and destination to stdout

Raises

urllib.error.HTTPError – If the url was unable to be downloaded

Returns

The path to the downloaded content

Return type

str

relenv.common.**extract_archive**(*to_dir*, *archive*)

Extract an archive to a specific location.

Parameters

- **to_dir** (*str*) – The directory to extract to
- **archive** (*str*) – The archive to extract

relenv.common.**fetch_url**(*url*, *fp*, *backoff=3*, *timeout=30*)

Fetch the contents of a url.

This method will store the contents in the given file like object.

relenv.common.**format_shebang**(*python*, *tpl='#!/bin/sh\n"true" \\\\\n"exec" "$(dirname "$(readlink -f "$0")")/{}" "$0" "$@"\n\\\\\n'*)

Return a formatted shebang.

relenv.common.**get_download_location**(*url*, *dest*)

Get the full path to where the url will be downloaded to.

Parameters

- **url** (*str*) – The url to donwload
- **dest** (*str*) – Where to download the url to

Returns

The path to where the url will be downloaded to

Return type

str

relenv.common.**get_toolchain**(*arch=None*, *root=None*)

Get a the toolchain directory, specific to the arch if supplied.

Parameters

- **arch** (*str*) – The architecture to get the toolchain for

- **root** (*str*) – The root of the relenv working directories to search in

Returns
The directory holding the toolchain

Return type
pathlib.Path

relenv.common.**get_triplet**(*machine=None*, *plat=None*)

Get the target triplet for the specified machine and platform.

If any of the args are None, it will try to deduce what they should be.

Parameters

- **machine** (*str*) – The machine for the triplet
- **plat** (*str*) – The platform for the triplet

Raises
RelenvException – If the platform is unknown

Returns
The target triplet

Return type
str

relenv.common.**list_archived_builds**()

Return a list of version, architecture and platforms for builds.

relenv.common.**plat_from_triplet**(*plat*)

Convert platform from build to the value of sys.platform.

relenv.common.**relative_interpreter**(*root_dir*, *scripts_dir*, *interpreter*)

Return a relativized path to the given scripts_dir and interpreter.

relenv.common.**runcmd**(**args*, ***kwargs*)

Run a command.

Run the provided command, raising an Exception when the command finishes with a non zero exit code. Arguments are passed through to `subprocess.run`

Returns
The process result

Return type
subprocess.CompletedProcess

Raises
RelenvException – If the command finishes with a non zero exit code

relenv.common.**sanitize_sys_path**(*sys_path_entries*)

Sanitize *sys.path* to only include paths relative to the onedir environment.

relenv.common.**work_dir**(*name*, *root=None*)

Get the absolute path to the relenv working directory of the given name.

Parameters

- **name** (*str*) – The name of the directory
- **root** (*str*) – The root directory that this working directory will be relative to

Returns
An absolute path to the requested relenv working directory

Return type
pathlib.Path

relenv.common.**work_dirs**(*root=None*)

Returns a WorkDirs instance based on the given root.

Parameters
root (*str*) – The desired root of relenv's working directories

Returns
A WorkDirs instance based on the given root

Return type
relenv.common.WorkDirs

relenv.common.**work_root**(*root=None*)

Get the root directory that all other relenv working directories should be based on.

Parameters
root (*str*) – An explicitly requested root directory

Returns
An absolute path to the relenv root working directory

Return type
pathlib.Path

2.7.3 Create

The `relenv create` command.

exception relenv.create.**CreateException**

Raised when there is an issue creating a new relenv environment.

relenv.create.**chdir**(*path*)

Context manager that changes to the specified directory and back.

Parameters
path (*str*) – The path to temporarily change to

relenv.create.**create**(*name*, *dest=None*, *arch=None*, *version=None*)

Create a relenv environment.

Parameters

- **name** (*str*) – The name of the environment
- **dest** (*str*) – The path the environment should be created under
- **arch** (*str*) – The architecture to create the environment for

Raises
CreateException – If there is a problem in creating the relenv environment

relenv.create.**main**(*args*)

The entrypoint into the `relenv create` command.

Parameters
args (*argparse.Namespace*) – The args passed to the command

relenv.create.**setup_parser**(*subparsers*)

Setup the subparser for the `create` command.

Parameters

subparsers (*argparse._SubParsersAction*) – The subparsers object returned from `add_subparsers`

2.7.4 Fetch

The `relenv fetch` command.

relenv.fetch.**main**(*args*)

The entrypoint into the `relenv fetch` command.

Parameters

args (*argparse.Namespace*) – The args passed to the command

relenv.fetch.**setup_parser**(*subparsers*)

Setup the subparser for the `fetch` command.

Parameters

subparsers (*argparse._SubParsersAction*) – The subparsers object returned from `add_subparsers`

2.7.5 Relocate

A script to ensure the proper rpaths are in place for the relenv environment.

relenv.relocate.**handle_elf**(*path*, *libs*, *rpath_only*, *root=None*)

Handle the parsing and pathcing of an ELF file.

Parameters

- **path** (*str*) – The path of the ELF file
- **libs** (*str*) – The libs directory
- **rpath_only** (*bool*) – If true, only ensure the correct rpaths are present and don't copy the file
- **root** (*str, optional*) – The directory to ensure the file is under, defaults to None

relenv.relocate.**handle_macho**(*path*, *root_dir*, *rpath_only*)

Ensure the given macho file has the correct rpath and is in th correct location.

Parameters

- **path** (*str*) – The path to a macho file
- **root_dir** (*str*) – The directory the file needs to reside under
- **rpath_only** (*bool*) – If true, only ensure the correct rpaths are present and don't copy the file

Returns

The information from `parse_macho` on the macho file.

relenv.relocate.**is_elf**(*path*)

Determines whether the given file is an ELF file.

Parameters
path (*str*) – The path to the file to check

Returns
Whether the file is an ELF file

Return type
bool

relenv.relocate.**is_in_dir**(*filepath*, *directory*)

Determines whether a file is contained within a directory.

Parameters

- **filepath** (*str*) – The path to the file to check
- **directory** (*str*) – The directory to check within

Returns
Whether the file is contained within the given directory

Return type
bool

relenv.relocate.**is_macho**(*path*)

Determines whether the given file is a macho file.

Parameters
path (*str*) – The path to the file to check

Returns
Whether the file is a macho file

Return type
bool

relenv.relocate.**main**(*root*, *libs_dir=None*, *rpath_only=True*, *log_level='DEBUG'*)

The entrypoint into the relocate script.

Parameters

- **root** (*str*) – The root directory to operate traverse for files to be patched
- **libs_dir** (*str, optional*) – The directory to place the libraries in, defaults to None
- **rpath_only** (*bool, optional*) – If true, only ensure the correct rpaths are present and don't copy the file, defaults to True
- **log_level** (*str, optional*) – The level to log at, defaults to "INFO"

relenv.relocate.**parse_macho**(*path*)

Run `otool -l <path>` and return its parsed output.

Parameters
path (*str*) – The path to the file

Returns
The parsed relevant RPATH content, or None if it isn't an object file

Return type
dict or None

relenv.relocate.**parse_otool_l**(*stdout*)

Parse the output of `otool -l <path>`.

Parameters
stdout (*str*) – The output of the `otool -l <path>` command

Returns
The parsed relevant output with command keys and path values

Return type
dict

relenv.relocate.**parse_readelf_d**(*stdout*)

Parse the output of `readelf -d <path>`.

Parameters
stdout (*str*) – The output of the `readelf -d <path>` command

Returns
The RPATH values

Return type
list

relenv.relocate.**parse_rpath**(*path*)

Run `readelf -d <path>` and return its parsed output.

Parameters
path (*str*) – The path to the file

Returns
The RPATH's found.

Return type
list

relenv.relocate.**patch_rpath**(*path*, *new_rpath*, *only_relative=True*)

Patch the rpath of a given ELF file.

Parameters

- **path** (*str*) – The path to an ELF file
- **new_rpath** (*str*) – The new rpath to add
- **only_relative** (*bool, optional*) – Whether or not to remove non-relative rpaths, defaults to True

Returns
The new rpath, or False if patching failed

Return type
str or bool

2.7.6 Runtime

This code is run when initializing the python interperter in a Relenv environment.

- Point Relenv's Openssl to the system installed Openssl certificate path
- Make sure pip creates scripts with a shebang that points to the correct python using a relative path.
- On linux, provide pip with the proper location of the Relenv toolchain gcc. This ensures when using pip any c dependencies are compiled against the proper glibc version.

class relenv.runtime.**RelenvImporter**(*wrappers=None*, *_loads=None*)

Handle runtime wrapping of module methods.

create_module(*spec*)

Create the module via a spec.

exec_module(*module*)

Exec module noop.

find_module(*module_name*, *package_path=None*)

Find modules being imported.

load_module(*name*)

Load an imported module.

class relenv.runtime.**TARGET**

Container for global pip target state.

class relenv.runtime.**Wrapper**(*module*, *wrapper*, *matcher='equals'*, *_loading=False*)

Wrap methods of an imported module.

matches(*module*)

Check if wrapper metches module being imported.

relenv.runtime.**bootstrap**()

Bootstrap the relenv environment.

relenv.runtime.**common**()

Late import relenv common.

relenv.runtime.**debug**(*string*)

Prints the provided message if RELENV_DEBUG is truthy in the environment.

Parameters

string (*str*) – The message to print

relenv.runtime.**finalize_options_wrapper**(*func*)

Wrapper around build_ext.finalize_options.

Used to add the relenv environment's include path.

relenv.runtime.**get_config_var_wrapper**(*func*)

Return a wrapper to resolve paths relative to the relenv root.

relenv.runtime.**get_config_vars_wrapper**(*func*, *mod*)

Return a wrapper to resolve paths relative to the relenv root.

relenv.runtime.**get_major_version**()

Current python major version.

relenv.runtime.**get_paths_wrapper**(*func*, *default_scheme*)

Return a wrapper to resolve paths relative to the relenv root.

relenv.runtime.**install_cargo_config**()

Setup cargo config.

relenv.runtime.**install_legacy_wrapper**(*func*)

Wrap pip's legacy install function.

This method determines any newly installed files and checks their RPATHs.

relenv.runtime.**install_wheel_wrapper**(*func*)

Wrap pip's wheel install function.

This method determines any newly installed files and checks their RPATHs.

relenv.runtime.**load_openssl_provider**(*name*)

Load an openssl module.

relenv.runtime.**path_import**(*name*, *path*)

Import module from a path.

This causes hashlib to be imported because of importing importlib.util so it can not be used until after openssl has been configured.

relenv.runtime.**pushd**(*new_dir*)

Changedir context.

relenv.runtime.**relenv_root**()

Return the relenv module root.

relenv.runtime.**relocate**()

Late import relenv relocate.

relenv.runtime.**set_env_if_not_set**(*name*, *value*)

Set an environment variable if not already set.

If the environment variable is already set and not equal to value, warn the user.

relenv.runtime.**set_openssl_modules_dir**(*path*)

Set the default search location for openssl modules.

relenv.runtime.**setup_crossroot**()

Setup cross root if needed.

relenv.runtime.**setup_openssl**()

Configure openssl certificate locations.

relenv.runtime.**wrap_cmd_install**(*name*)

Wrap pip install command to store target argument state.

relenv.runtime.**wrap_distutils_command**(*name*)

distutils.command wrapper.

relenv.runtime.**wrap_locations**(*name*)

Wrap pip locations to fix locations when installing with target.

relenv.runtime.**wrap_pip_build_wheel**(*name*)

pip._internal.operations.build wrapper.

relenv.runtime.**wrap_pip_distlib_scripts**(*name*)

pip.distlib.scripts wrapper.

relenv.runtime.**wrap_pip_install_legacy**(*name*)

pip._internal.operations.install.legacy wrapper.

relenv.runtime.**wrap_pip_install_wheel**(*name*)

pip._internal.operations.install.wheel wrapper.

relenv.runtime.**wrap_req_command**(*name*)

Honor ignore installed option from pip cli.

relenv.runtime.**wrap_sysconfig**(*name*)

Sysconfig wrapper.

relenv.runtime.**wrapsitecustomize**(*func*)

Wrap site.execsitecustomize.

This allows relenv a hook to be the last thing that runs when pythong is setting itself up.

2.7.7 Toolchain

The `relenv toolchain` command.

relenv.toolchain.**build**(*arch*, *dirs*, *machine*, *ctngdir*)

Build a toolchaing for the given arch.

Parameters

- **arch** (*str*) – The architecture to build for
- **dirs** (`relenv.common.WorkDirs`) – The working directories
- **machine** (*str*) – The machine to build for
- **ctngdir** (`pathlib.Path`) – The directory holding crosstool-ng

relenv.toolchain.**fetch**(*arch*, *toolchain*, *clean=False*, *version='0.13.11'*)

Fetch a toolchain and extract it to the filesystem.

Parameters

- **arch** (*str*) – The architecture of the toolchain
- **toolchain** (*str*) – Where to extract the toolchain
- **clean** (*bool*) – If true, clean the toolchain directories first

relenv.toolchain.**main**(*args*)

The entrypoint into the `relenv toolchain` command.

Parameters

args (`argparse.Namespace`) – The arguments for the command

relenv.toolchain.**setup_parser**(*subparsers*)

Setup the subparser for the `toolchain` command.

Parameters

subparsers (*argparse._SubParsersAction*) – The subparsers object returned from `add_subparsers`

2.7.8 Build

The build command consists of a few different flows depending on the platform being worked on.

Common code can be found *here*.

Common Build Operations

Build process common methods.

class relenv.build.common.**Builder**(*root=None*, *recipies=None*, *build_default=<function build_default>*, *populate_env=<function populate_env>*, *force_download=False*, *arch='x86_64'*, *version=''*)

Utility that handles the build process.

Parameters

- **root** (*str*) – The root of the working directories for this build
- **recipies** – The instructions for the build steps
- **build_default** (*types.FunctionType*) – The default build function, defaults to `build_default`
- **populate_env** (*types.FunctionType*) – The default function to populate the build environment, defaults to `populate_env`
- **force_download** (*bool*) – If True, forces downloading the archives even if they exist, defaults to False
- **arch** (*str*) – The architecture being built

add(*name*, *build_func=None*, *wait_on=None*, *download=None*)

Add a step to the build process.

Parameters

- **name** (*str*) – The name of the step
- **build_func** (*types.FunctionType, optional*) – The function that builds this step, defaults to None
- **wait_on** (*list, optional*) – Processes to wait on before running this step, defaults to None
- **download** (*dict, optional*) – A dictionary of download information, defaults to None

build(*steps=None*, *cleanup=True*)

Build!

Parameters

- **steps** (*list, optional*) – The steps to run, defaults to None
- **cleanup** (*bool, optional*) – Whether to clean up or not, defaults to True

check_prereqs()

Check pre-requsists for build.

This method verifies all requrements for a successful build are satisfied.

Returns

Returns a list of string describing failed checks

Return type
list

`clean`()

Completely clean up the remnants of a relenv build.

`cleanup`()

Clean up the build directories.

`download_files`(*steps=None*, *force_download=False*)

Download all of the needed archives.

Parameters
steps (*list, optional*) – The steps to download archives for, defaults to None

`run`(*name*, *event*, *build_func*, *download*)

Run a build step.

Parameters

- **name** (*str*) – The name of the step to run
- **event** (`multiprocessing.Event`) – An event to track this process' status and alert waiting steps
- **build_func** (*types.FunctionType*) – The function to use to build this step
- **download** (`Download`) – The `Download` instance for this step

Returns
The output of the build function

`set_arch`(*arch*)

Set the architecture for the build.

Parameters
arch (*str*) – The arch to build

`class relenv.build.common.Builds`

Collection of builds.

`class relenv.build.common.Dirs`(*dirs*, *name*, *arch*, *version*)

A container for directories during build time.

Parameters

- **dirs** (`relenv.common.WorkDirs`) – A collection of working directories
- **name** (*str*) – The name of this collection
- **arch** (*str*) – The architecture being worked with

`to_dict`()

Get a dictionary representation of the directories in this collection.

Returns
A dictionary of all the directories

Return type
dict

class relenv.build.common.**Download**(*name*, *url*, *fallback_url=None*, *signature=None*, *destination=''*, *version=''*, *md5sum=None*, *checkfunc=None*, *checkurl=None*)

A utility that holds information about content to be downloaded.

Parameters

- **name** (*str*) – The name of the download
- **url** (*str*) – The url of the download
- **signature** (*str*) – The signature of the download, defaults to None
- **destination** (*str*) – The path to download the file to
- **version** (*str*) – The version of the content to download
- **md5sum** (*str*) – The md5 sum of the download

exists()

True when the artifact already exists on disk.

Returns
True when the artifact already exists on disk

Return type
bool

fetch_file()

Download the file.

Returns
The path to the downloaded content, and whether it was downloaded.

Return type
tuple(str, bool)

fetch_signature(*version*)

Download the file signature.

Returns
The path to the downloaded signature.

Return type
str

static validate_md5sum(*archive*, *md5sum*)

True when when the archive matches the md5 hash.

Parameters

- **archive** (*str*) – The path to the archive to validate
- **md5sum** (*str*) – The md5 sum to validate against

Returns
True if the sums matched, else False

Return type
bool

static validate_signature(*archive*, *signature*)

True when the archive's signature is valid.

Parameters

- **archive** (*str*) – The path to the archive to validate
- **signature** (*str*) – The path to the signature to validate against

Returns
True if it validated properly, else False

Return type
bool

relenv.build.common.**all_dirs**(*root*, *recurse=True*)

Get all directories under and including the given root.

Parameters

- **root** (*str*) – The root directory to traverse
- **recurse** (*bool, optional*) – Whether to recursively search for directories, defaults to True

Returns
A list of directories found

Return type
list

relenv.build.common.**build_default**(*env*, *dirs*, *logfp*)

The default build function if none is given during the build process.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.common.**build_openssl**(*env*, *dirs*, *logfp*, *fips=False*)

Build openssl.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.common.**build_sqlite**(*env*, *dirs*, *logfp*)

Build sqlite.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.common.**create_archive**(*tarfp*, *toarchive*, *globs*, *logfp=None*)

Create an archive.

Parameters

- **tarfp** (*file*) – A pointer to the archive to be created
- **toarchive** (*str*) – The path to the directory to archive

- **globs** (*list*) – A list of filtering patterns to match against files to be added
- **logfp** (*file*) – A pointer to the log file

relenv.build.common.**finalize**(*env*, *dirs*, *logfp*)

Run after we've fully built python.

This method enhances the newly created python with Relenv's runtime hacks.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.common.**find_sysconfigdata**(*pymodules*)

Find sysconfigdata directory for python installation.

Parameters
pymodules (*str*) – Path to python modules (e.g. lib/python3.10)

Returns
The name of the sysconig data module

Return type
str

relenv.build.common.**install_runtime**(*sitepackages*)

Install a base relenv runtime.

relenv.build.common.**install_sysdata**(*mod*, *destfile*, *buildroot*, *toolchain*)

Create a Relenv Python environment's sysconfigdata.

Helper method used by the *finalize* build method to create a Relenv Python environment's sysconfigdata.

Parameters

- **mod** (types.ModuleType) – The module to operate on
- **destfile** (*str*) – Path to the file to write the data to
- **buildroot** (*str*) – Path to the root of the build
- **toolchain** (*str*) – Path to the root of the toolchain

relenv.build.common.**patch_shebang**(*path*, *old*, *new*)

Replace a file's shebang.

Parameters

- **path** (*str*) – The path of the file to patch
- **old** (*str*) – The old shebang, will only patch when this is found
- **name** (*str*) – The new shebang to be written

relenv.build.common.**patch_shebangs**(*path*, *old*, *new*)

Traverse directory and patch shebangs.

Parameters

- **path** (*str*) – The of the directory to traverse
- **old** (*str*) – The old shebang, will only patch when this is found

- **name** (*str*) – The new shebang to be written

relenv.build.common.**print_ui**(*events*, *processes*, *fails*, *flipstat=None*)

Prints the UI during the relenv building process.

Parameters

- **events** (*dict*) – A dictionary of events that are updated during the build process
- **processes** (*dict*) – A dictionary of build processes
- **fails** (*list*) – A list of processes that have failed
- **flipstat** (*dict, optional*) – A dictionary of process statuses, defaults to {}

relenv.build.common.**verify_checksum**(*file*, *checksum*)

Verify the checksum of a files.

Parameters

- **file** (*str*) – The path to the file to check.
- **checksum** (*str*) – The checksum to verify against

Raises

RelenvException – If the checksum verification failed

Returns

True if it succeeded, or False if the checksum was None

Return type

bool

Building on Linux

The linux build process.

relenv.build.linux.**build_bzip2**(*env*, *dirs*, *logfp*)

Build bzip2.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_gdbm**(*env*, *dirs*, *logfp*)

Build gdbm.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_krb**(*env*, *dirs*, *logfp*)

Build kerberos.

Parameters

- **env** (*dict*) – The environment dictionary

- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_libffi**(*env*, *dirs*, *logfp*)

Build libffi.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_libxcrypt**(*env*, *dirs*, *logfp*)

Build libxcrypt.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_ncurses**(*env*, *dirs*, *logfp*)

Build ncurses.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_python**(*env*, *dirs*, *logfp*)

Run the commands to build Python.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**build_zlib**(*env*, *dirs*, *logfp*)

Build zlib.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.linux.**populate_env**(*env*, *dirs*)

Make sure we have the correct environment variables set.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories

Building on MacOS

The darwin build process.

relenv.build.darwin.**build_python**(*env*, *dirs*, *logfp*)

Run the commands to build Python.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.darwin.**populate_env**(*env*, *dirs*)

Make sure we have the correct environment variables set.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories

Building on Windows

The windows build process.

relenv.build.windows.**build_python**(*env*, *dirs*, *logfp*)

Run the commands to build Python.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.windows.**finalize**(*env*, *dirs*, *logfp*)

Finalize sitecustomize, relenv runtime, and pip for Windows.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories
- **logfp** (*file*) – A handle for the log file

relenv.build.windows.**populate_env**(*env*, *dirs*)

Make sure we have the correct environment variables set.

Parameters

- **env** (*dict*) – The environment dictionary
- **dirs** (relenv.build.common.Dirs) – The working directories

The following is for the entrypoint into the build command.

The `relenv build` command.

relenv.build.**main**(*args*)

The entrypoint to the `build` command.

Parameters
args (`argparse.Namespace`) – The arguments to the command

relenv.build.**platform_module**()

Return the right module based on *sys.platform*.

relenv.build.**platform_versions**()

Return the right module based on *sys.platform*.

relenv.build.**setup_parser**(*subparsers*)

Setup the subparser for the `build` command.

Parameters
subparsers (*argparse._SubParsersAction*) – The subparsers object returned from `add_subparsers`

2.8 License

2.8.1 Apache License

Version 2.0, January 2004 <`http://www.apache.org/licenses/ <http://www.apache.org/licenses/>`_>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- **(a)** You must give any other recipients of the Work or Derivative Works a copy of

this License; and

- **(b)** You must cause any modified files to carry prominent notices stating that You

changed the files; and

- **(c)** You must retain, in the Source form of any Derivative Works that You distribute,

all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

- **(d)** If the Work includes a "NOTICE" text file as part of its distribution, then any

Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works

that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Copyright (c) 2011-2022 VMware, Inc. All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

2.9 Changelog

2.9.1 0.13.11

- Add regression test for system fips module usage
- Fix fips module usage on photon os.

2.9.2 0.13.10

- Add a build-id for downstream rpm packaging

2.9.3 0.13.9

- Revert with-dbg flag on python builds.

2.9.4 0.13.8

- Fix wart in python-config's shebang cuasing syntax error.

2.9.5 0.13.7

- Load relenv's openssl legacy provider before setting modules dir to system location for the system's fips provider.

2.9.6 0.13.6

- Do not set openssl modules directory on windows since were still on 1.1.x
- Fix load module deprecations warnings
- Ignore load module imporet warnings for now

2.9.7 0.13.5

- Bump to Python 3.10.13 and 3.11.5 due to CVE-2023-40217 and CVE-2023-41105
- Include debug symbols to enable gdb debugging
- Set openssl module locations via c api rather than environment variable
- Default to the system's openssl modules directory
- Bump dependency versions

2.9.8 0.13.4

- Fix pip installing multiple packages with scripts to a target directory
- Finish bootstrap before importing hashlib so our openssl modules will be found.

2.9.9 0.13.3

- Upgrade openssl to 3.1.2

2.9.10 0.13.2

- Always use relenv's openssl modules directory

2.9.11 0.13.1

- Determine openssl modules directory at runtime

2.9.12 0.13.0

- Tests and fixes for installing m2crypto
- Fix pipelines to upload to repo.saltstack.io
- Ship with openssl 3.1.1 on linux and darwin for FIPS compatability
- Update openssl and python minior version to address CVEs

2.9.13 0.12.3

- Preserve ignore installed option when using pip with a target.

2.9.14 0.12.2

- Fix path comparison bug on win32

2.9.15 0.12.1

- Be more robust when getting system python config

2.9.16 0.12.0

- Add support building on M1 mac
- Fix wart in relenv create's help message
- Look in path for system python
- Provide sane defaults for pip when no system python is found
- Fix shebangs when using pip –target to install packages
- Fix uninstalling packages installed with pip –target

2.9.17 0.11.2

- Fetch files from repo.saltproject.io first.

2.9.18 0.11.1

- Import all relenv modules using a reletive path when relenv.runtime is imported.

2.9.19 0.11.0

- Use a pth file instead of sitecustomize for relenv runtime
- Fix errors in documentation
- Default to using system libraries, act more like virtualenv
- Source relenv buildenv instead of eval
- Upgrade XZ and SQLite
- Upgrade minor python versions (3.10.11 and 3.11.3)

2.9.20 0.10.1

- Fix bug in runtime.bootstrap on linux when no toolchain exists

2.9.21 0.10.0

- Add buildenv to support building of additional libraries
- Add check to support installation of additional libraries
- Add examples of building libgit2, open-ldap and libcurl

2.9.22 0.9.0

- Add support for rust c extensions
- Add sys.RELENV attribute to runtime bootstrap
- Fix ImportWarning thrown by RelenvImporter
- Refactor RelenvImporter

2.9.23 0.8.2

- Fix SHEBANG when installing scripts to root

2.9.24 0.8.1

- Fix bug in crypt module's rpath

2.9.25 0.8.0

- Better fix for rpaths of pip installed C extensions
- Fetch current version not 'latest'
- Add libxcrypt to linux builds
- Shellcheck script shebangs

2.9.26 0.7.0

- Update to python 3.10.10
- Remove C-python test suite from build
- Fix rpath on pip installed C moudles

2.9.27 0.6.0

- Add python 3.11.2
- Upgrade linux python depenencies
- Add version check script

2.9.28 0.5.0

- Add '–version' option to cli
- Support symlinks on OSes without coreutils installed

2.9.29 0.4.10

- Update windows python to 3.10.x

2.9.30 0.4.9

- Make shebangs in Python's modules relative.

2.9.31 0.4.8

- Statically link aarch64 toolchains for portability

2.9.32 0.4.7

- Wrap build_ext finalize_options method to add relenv include directory
- Add tests that installs m2crypto on linux

2.9.33 0.4.6

- Script shebangs now work when symlinked

2.9.34 0.4.5

- Build newest python release
- Do not define SSL_CERT_FILE when file does not exit
- Only define ssl environment variables if not already set

2.9.35 0.4.4

- Fix scripts relative to launcher_dir on windows using RELENV_PIP_DIR
- Add flake8 for linting

2.9.36 0.4.3

- Fix arch flag when fetching builds
- Cleanup changelog syntax
- Add test for virtual environments based on relenv environments

2.9.37 0.4.2

- General code clean up based on pylint results
- Fix virtualenvs created from relenvs
- The fetch and toolchain always show download urls and destinations
- Fix oldest supported Mac OS version (10.5)
- Docs improvements

2.9.38 0.4.1

- Work around issue on Mac where Python is linking to /usr/local Issue #46

2.9.39 0.4.0

- Fix issue where relenv runtime was being imported from user site packages
- Added test to install salt with USE_STATIC_PACAKGES environment set
- Switch CI/CD to use saltstack hosted runners
- General code cleanup

2.9.40 0.3.0

- The toolchain command defaults to the build box architecture
- Build macos on catalinia for now

2.9.41 0.2.1

- Fix 'RELENV_PIP_DIR' environment variable on python <= 3.10 (Windows)

2.9.42 0.2.0

- Skip downloads that exist and are valid.
- Inlude changelog in documentation.
- Better help when no sub-command given.
- Add some debuging or relocate module.

2.9.43 0.1.0

- Multiple fixes for cross compilation support.

2.9.44 0.0.3

- Build pipeline improvements.

2.9.45 0.0.2

- Fetch defaults to the latest version of pre-built Python build.
- Build and test pipeline improvements
- Add package description

2.9.46 0.0.1

- Initial release of Relenv. Build relocatable python builds for Linux, Macos and Windows.

Relenv has no public API, but you can find documentation for the internals of relenv *here*.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

H

I

L

M

P

R

S

T

V

W